AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A method for dynamically configuring selected methods for instrument-based profiling at run-time, comprising:

identifying a root method in a target application, wherein only methods that are reachable from the root method during execution of the target application are to be instrumented;

instrumenting the root method after it is identified, wherein instrumenting the root method involves dynamically patching the root method while the target application is executing; and

upon executing a given instrumented method, determining if the given instrumented method is about to be executed for the first time, and if so, instrumenting methods that,

are called by the given instrumented method,
are loaded, and
have not been instrumented before.

2. (Original) The method of claim 1, wherein identifying the root method involves allowing a user to specify the root method.

3. (Original) The method of claim 1, wherein determining if the given instrumented method is about to be executed for the first time involves executing instrumentation code within the given instrumented method, wherein the

instrumentation code is executed whenever the given instrumented method is called, and is executed before any other instructions of the given instrumented method are executed.

4. (Currently amended) The method of claim 3,

wherein the instrumentation code checks a global executed-once-or-more flag associated with the given instrumented method, which is initially set to false;

wherein if the executed-once-or-more flag is false, the instrumentation code knows that the given instrumented method has not been called before, so it performs instrumentation operations as necessary and sets ~~the~~ an instrumented once or more flag to true; and

wherein if the executed-once-or-more flag ~~is true~~ is true, the instrumentation code does not perform instrumentation operations.

5. (Currently amended) The method of claim 1, wherein if a call to a virtual method is encountered in a given instrumented method that is about to be executed for the first time, the method further comprises:

identifying a class for the virtual method based upon an object type associated with ~~the~~ a call site; and

instrumenting methods corresponding to the virtual method in the identified class and in associated subclasses.

6. (Original) The method of claim 5, wherein if the identified class does not have an implementation of the virtual method, the method additionally involves identifying a nearest superclass of the identified class that has an implementation of the virtual method.

7. (Original) The method of claim 1, wherein instrumenting a method involves dynamically patching the method while the target application is executing.

8. (Original) The method of claim 1, wherein instrumenting a method involves inserting profiling instrumentation code into the method, wherein the profiling instrumentation code includes:

method entry code that takes a first time measurement at the beginning of a method;

method exit code that takes a second time measurement at the end of the method; and

wherein the first and second time measurements are used to calculate an execution time for the method.

9. (Original) The method of claim 8, wherein the method entry code determines if the given instrumented method is about to be executed for the first time.

10. (Original) The method of claim 1, wherein the tasks of identifying methods and instrumenting methods are performed by a remote profiler client that communicates with a virtual machine executing the target application.

11. (Currently amended) The method of claim 1, wherein code that makes up the target application includes platform-independent ~~Java~~ bytecodes.

12. (Currently amended) A computer-readable storage medium storing instructions that when executed by a computer cause the computer to perform a method for dynamically configuring selected methods for instrument-based

profiling at run-time, wherein the computer-readable storage medium does not include computer instruction signals embodied in a transmission medium, and wherein the method comprises~~ing~~:

identifying a root method in a target application, wherein only methods that are reachable from the root method during execution of the target application are to be instrumented;

instrumenting the root method after it is identified, wherein instrumenting the root method involves dynamically patching the root method while the target application is executing; and

upon executing a given instrumented method, determining if the given instrumented method is about to be executed for the first time, and if so, instrumenting methods that,

are called by the given instrumented method,

are loaded, and

have not been instrumented before.

13. (Original) The computer-readable storage medium of claim 12, wherein identifying the root method involves allowing a user to specify the root method.

14. (Original) The computer-readable storage medium of claim 12, wherein determining if the given instrumented method is about to be executed for the first time involves executing instrumentation code within the given instrumented method, wherein the instrumentation code is executed whenever the given instrumented method is called, and is executed before any other instructions of the given instrumented method are executed.

15. (Currently amended) The computer-readable storage medium of claim 14,

wherein the instrumentation code checks a global executed-once-or-more flag associated with the given instrumented method, which is initially set to false;

wherein if the executed-once-or-more flag is false, the instrumentation code knows that the given instrumented method has not been called before, so it performs instrumentation operations as necessary and sets ~~the~~ an instrumented once or more flag to true; and

wherein if the executed-once-or-more flag ~~is true~~ is true, the instrumentation code does not perform instrumentation operations.

16. (Currently amended) The computer-readable storage medium of claim 12, wherein if a call to a virtual method is encountered in a given instrumented method that is about to be executed for the first time, the method further comprises:

identifying a class for the virtual method based upon an object type associated with ~~the~~ a call site; and

instrumenting methods corresponding to the virtual method in the identified class and in associated subclasses.

17. (Original) The computer-readable storage medium of claim 16, wherein if the identified class does not have an implementation of the virtual method, the method additionally involves identifying a nearest superclass of the identified class that has an implementation of the virtual method.

18. (Original) The computer-readable storage medium of claim 12, wherein instrumenting a method involves dynamically patching the method while the target application is executing.

19. (Original) The computer-readable storage medium of claim 12, wherein instrumenting a method involves inserting profiling instrumentation code into the method, wherein the profiling instrumentation code includes:

method entry code that takes a first time measurement at the beginning of a method;

method exit code that takes a second time measurement at the end of the method; and

wherein the first and second time measurements are used to calculate an execution time for the method.

20. (Original) The computer-readable storage medium of claim 19, wherein the method entry code determines if the given instrumented method is about to be executed for the first time.

21. (Original) The computer-readable storage medium of claim 12, wherein the tasks of identifying methods and instrumenting methods are performed by a remote profiler client that communicates with a virtual machine executing the target application.

22. (Currently amended) The computer-readable storage medium of claim 12, wherein code that makes up the target application includes platform-independent ~~Java~~ bytecodes.

23. (Currently amended) An apparatus for dynamically configuring selected methods for instrument-based profiling at run-time, comprising:

~~an identification mechanism~~a first circuit within a processor configured to identify a root method in a target application, wherein only methods that are reachable from the root method during execution of the target application are to be instrumented; and

~~an instrumentation mechanism~~a second circuit within the processor configured to instrument the root method after it is identified, wherein while instrumenting the root method, the second circuit is configured to dynamically patch the root method while the target application is executing;

wherein when a given instrumented method is executed, the ~~instrumentation mechanism~~second circuit is configured to determine if the given instrumented method is about to be executed for the first time, and if so, to instrument methods that,

are called by the given instrumented method,

are loaded, and

have not been instrumented before.

24. (Currently amended) The apparatus of claim 23, wherein the ~~identification mechanism~~first circuit is configured to identify the root method by allowing a user to specify the root method.

25. (Currently amended) The apparatus of claim 23, wherein while determining if the given instrumented method is about to be executed for the first time, the ~~instrumentation mechanism~~second circuit is configured to execute instrumentation code within the given instrumented method, wherein the instrumentation code is executed whenever the given instrumented method is called, and is executed before any other instructions of the given instrumented method are executed.

26. (Currently amended) The apparatus of claim 25,

wherein the instrumentation code checks a global executed-once-or-more flag associated with the given instrumented method, which is initially set to false;

wherein if the executed-once-or-more flag is false, the instrumentation code knows that the given instrumented method has not been called before, so it performs instrumentation operations as necessary and sets ~~the~~ an instrumented once or more flag to true; and

wherein if the executed-once-or-more flag ~~is true~~ is true, the instrumentation code does not perform instrumentation operations.

27. (Currently amended) The apparatus of claim 23, wherein if a call to a virtual method is encountered in a given instrumented method that is about to be executed for the first time, the ~~instrumentation mechanism~~second circuit is configured to:

identify a class for the virtual method based upon an object type associated with ~~the~~ a call site; and to

instrument methods corresponding to the virtual method in the identified class and in associated subclasses.

28. (Currently amended) The apparatus of claim 27, wherein if the identified class does not have an implementation of the virtual method, the ~~instrumentation mechanism~~second circuit is configured to identify a nearest superclass of the identified class that has an implementation of the virtual method.

29. (Currently amended) The apparatus of claim 23, wherein the ~~instrumentation mechanism~~second circuit is configured to instrument a method by dynamically patching the method while the target application is executing.

30. (Currently amended) The apparatus of claim 23, wherein the ~~instrumentation mechanism~~second circuit is configured to instrument a method by inserting profiling instrumentation code into the method, wherein the profiling instrumentation code includes:

method entry code that takes a first time measurement at the beginning of a method;

method exit code that takes a second time measurement at the end of the method; and

wherein the first and second time measurements are used to calculate an execution time for the method.

31. (Original) The apparatus of claim 30, wherein the method entry code determines if the given instrumented method is about to be executed for the first time.

32. (Currently amended) The apparatus of claim 23, wherein the ~~identification mechanism~~first circuit and the ~~instrumentation mechanism~~second circuit are located within a remote profiler client that communicates with a virtual machine executing the target application.

33. (Currently amended) The apparatus of claim 23, wherein code that makes up the target application includes platform-independent ~~Java~~ bytecodes.